Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Anej Placer

Generiranje glasbenih sekvenc z uporabo modela umetnega življenja

diplomsko delo

univerzitetni študijski program prve stopnje
računalništvo in informatika

Mentor: doc. dr. Iztok Lebar Bajec
Somentor: doc. dr. Matija Marolt

Ljubljana 2014

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

V diplomski nalogi postavite model umetnega življenja, ki bo sposoben generiranja "poslušljivih" glasbenih sekvenc. Poslušljivost glasbene sekvence ocenite z vidika glasbene teorije, kjer upoštevajte vidike konsonance in disonance, aktivnosti in neaktivnosti, ter melodične in ritmične raznolikosti. Posamezen agent, kot enota umetnega življenja naj se celote - glasbene sekvence, ki nastane kot plod generiranja tonov s strani množice posameznikov - ne zaveda, njegovi odzivi naj bodo pogojeni na lokalno dogajanje v nekem trenutku izvajanja glasbene sekvence. Za model umetnega življenja pripravite tudi grafični uporabniški vmesnik, ki bo uporabniku dopuščal upravljanje z začetnimi in vsemi ostalimi parametri.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Anej Placer, z vpisno številko **63080315**, sem avtor diplomskega dela z naslovom:

Generiranje glasbenih sekvenc z uporabo modela umetnega življenja

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Iztoka Lebarja Bajca in somentorstvom doc. dr. Matije Marolta,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 17. september 2014 Podpis avtorja:

Zahvaljujem se za vso podporo pri odpravi v neznane vode z neznanim izidom.

Rin.

Kazalo

Povzetek

Abstract

1 Uvod **1**

2 Orodja **3**

2.1 Qt 3

2.2 Boost 3

2.3 MIDI 4

3 Glasbena teorija **9**

3.1 Intervali 9

3.2 Ritem 12

4 Struktura aplikacije **13**

4.1 Okolje 13

4.2 Entiteta 13

4.3 Odziv 14

4.4 Generator naključnih števil 17

4.5 Poganjalnik MIDI 17

4.6 Grafični uporabniški vmesnik 18

4.7 Jedro 19

5 Algoritem **23**

5.1 Cilj 23

5.2 Iskanje rešitve 23

5.3 Ocena 24

6 Uporaba in rezultati **29**

6.1 Začetni parametri 29

6.2 Strategija iskanja 31

6.3 Rezultati 32

7 Sklepne ugotovitve **33**

7.1 Pristop 33

7.2 Uporabnost 33

7.3 Nadgradnja 34

Povzetek

Diplomsko delo opisuje način generiranja glasbenih sekvenc s pomočjo modela avtonomnih agentov. Agenti (entitete) kot odziv na dogodke iz okolice oddajajo določene glasbene tone. Posamezna entiteta je sposobna različnih odzivov in njihovo sosledje z odzivi ostalih entitet tvori glasbene sekvence. Entitete se celote ne zavedajo. Po vsaki izvedbi določenega števila korakov ocenitvena funkcija, ki narekuje obliko iskane rešitve, oceni prispevek posameznika k celoti. Ocena posamezne entitete je sestavljena iz delnih ocen različnih glasbenih vidikov producirane sekvence. Določa verjetnost mutacije parametrov posameznih odzivov nove generacije entitete. Ustvarjen algoritem torej z uporabo usmerjenega iskanja išče vrednosti parametrov in zaporedje odzivov, ki jih morajo entitete izvesti, da bodo v primeru množice entitet producirane sekvence sestavljale poslušljivo celoto.

Ključne besede: umetno življenje, model avtonomnih agentov, glasba, generiranje glasbenih sekvenc, ocenjevanje glasbenih sekvenc, C++, MIDI, Qt.

Abstract

This BSc thesis describes the means of generating music sequences using an agent-based model. As a response to surrounding events, agents (entities) emit certain musical tones. Individual entity is capable of several different responses, which in combination with responses of the remaining entities form musical sequences. Entities aren't aware of the end goal. After every execution of a certain number of steps, their contribution is assessed by a fitness function, which dictates the shape of the desired solution. The score of the individual entity consists of partial assessments of different musical aspects of the produced sequence. It determines the chance of mutation of the responses' parameters of a new generation of an entity. Created directional search algorithm searches for parameter values and sequence of responses that the entities must perform, so that in the case of a set of entities the produced sequences would form a coherent whole.

Keywords: artificial life, agent-based model, music, music sequence generation, music sequence evaluation, C++, MIDI, Qt.

Poglavje 1

Uvod

Svet glasbe je svet idej, ki je na trenutke težko dostopen. Glasbeniki se zato poslužujejo najrazličnejših pristopov in orodij, ki bi jim lahko odprli nova obzorja. Kot orodje je primeren seveda tudi računalnik. Obstaja vrsta algoritmičnih pristopov h glasbeni kompoziciji [1]. Ob tem se hitro porodi vprašanje, kakšne vrste algoritmov so sploh primerne za dano nalogo? Izkorišča se genetske algoritme [2], gramatike [3], ekspertne sisteme in matematične modele [4]. Uporabljani so tudi algoritmi, kot so fraktali in L-sistemi [5]. Kaj pa umetno življenje?

Cilj diplomske naloge je generiranje smiselnih in raznolikih glasbenih sekvenc s pomočjo umetnega življenja. Predstavljen je model avtonomnih agentov (angl. *agent-based model*), katerega posamezniki (entitete) z različnimi odzivi na dogodke iz okolice oddajajo tone MIDI in skupaj tvorijo glasbene sekvence. Glasbeno sekvenco ustvari množica entitet s točno določenimi parametri. Parametri se z mutacijo skozi generacije spreminjajo, kar pripelje do novih sekvenc. Za ocenjevanje glasbe je potrebna definicija vidikov poslušljivosti. Opisana ocenitvena funkcija ocenjuje poslušljivost ustvarjenih sekvenc, njihova ocena pa vpliva na evolucijo entitet.

Zelo aktiven na področju računalniško generirane glasbe je brazilski skladatelj in znanstvenik prof. dr. Eduardo Reck Miranda. Ukvarja se predvsem z umetno inteligenco v glasbi s pomočjo katere poskuša pripraviti računalniški program do kompozicije originalne glasbe. Z nevronskimi mrežami je uspel imitirati določene glasbene žanre, cilj pravih, popolnoma originalnih kompozicij pa za enkrat ostaja nedosežen. V članku [3] opisuje tri modele avtonomnih agentov. Pri prvem agenti

drug drugemu ocenjujejo zaigrane sekvence, pri drugem jih med seboj primerjajo in izboljšujejo in pri tretjem ustvarjajo gramatike za kompozicijo glasbe, ki odraža njihova čustva. V članku [6] poleg modelov avtonomnih agentov opisuje tudi pristop h generiranju glasbe s pomočjo celularnih avtomatov. Njegova dela [7] so priporočena v branje vsem, ki jih zanima področje računalniško generirane glasbe. Kot dobra zbirka poglavij s tematiko umetne inteligence in glasbe je priporočena v branje knjiga [8].

Poglavje 2

Orodja

2.1 Qt

Qt je prosto in odprto-kodno ogrodje, ki deluje v različnih okoljih. Kot knjižnica gradnikov se uporablja predvsem za izdelavo aplikacij z grafičnim vmesnikom. Primerno je tudi za izdelavo programov brez grafičnega vmesnika, kot so strežniki in konzole.

Qt uporablja standarden, s posebnim pred-procesorjem obogaten, programski jezik *C++*. Preko povezovalnih vmesnikov ga je moč uporabiti tudi v drugih programskih jezikih. Deluje na vseh glavnih operacijskih sistemih, kot tudi na nekaterih mobilnih. Poleg gradnikov za grafične vmesnike ponuja tudi podporo za dostop do podatkovnih zbirk (*SQL*), razčlenjevalnik *XML*, upravljanje z nitmi, omrežno podporo in poenoten *API* za upravljanje z datotekami na različnih operacijskih sistemih. Na voljo pod komercialno licenco, GPL v3 in LGPL v2.

Znan je predvsem po uporabi v programih, kot so *Wolfram Mathematica*, *Google Earth*, *KDE*, medijski predvajalnik *VLC*, *Skype*, *Ubuntu* in *Autodesk Maya* [9].

2.2 Boost

Boost je zbirka knjižnic za programski jezik *C++*. Ponuja podporne strukture in mehanizme za naloge, kot so generiranje pseudo-naključnih števil, nitenje, linearna algebra, procesiranje slik, regularni izrazi in testiranje enot [10].

2.3 MIDI

MIDI (*Musical Instrument Digital Interface*) je tehnični standard, ki opisuje protokol, digitalni vmesnik in povezovalnike. Digitalnim instrumentom, računalnikom in podobnim napravam omogoča, da med seboj komunicirajo z *dogodki MIDI*. Namenjen je hrambi ter prenosu glasbe v digitalni obliki [11].

Za razliko od ostalih digitalnih audio formatov (*.wav*, *.mp3*, *.aiff* itd.) MIDI ne hrani posnetih zvokov. Datoteko MIDI sestavlja vrsta ukazov, ki določajo notacijo, višino tona, dinamiko in posebne zvočne učinke. Ti ukazi predvajalni napravi opišejo, kako ustvariti želene zvoke.

Prednosti formata MIDI so kompaktnost (v primerjavi z ostalimi formati), enostavnost urejevanja in izbira želenega instrumenta.

V operacijskem sistemu *Windows* se lahko za oddajanje zvokov MIDI uporablja *Windows API* (*Windows multimedia* - mmsystem.h).

2.3.1 Struktura

Format MIDI je sestavljen iz dveh delov - glave (tip *MThd*) in ene ali več sledi (tip *MTrk*) [12]. Struktura posameznega dela je prikazana v Tabeli 2.1.

`tip`	`dolžina`	`podatki`
4 bajti	4 bajti	*dolžina* bajtov

Tabela 2.1: Struktura posameznega dela.

Glava

Podatkovni del glave vsebuje tri 16-bitna polja, ki določajo format, število sledi in ritem.

Standardna dolžina podatkovnega dela je 6 bajtov. Od programa, ki bere datoteko MIDI je zaradi možnih razširitev zahtevano, da upošteva polje za dolžino podatkov, čeprav je lahko vrednost večja, kot je pričakovano. Podatki nepričakovane velikosti se tako ignorirajo. Struktura glave je prikazana v Tabeli 2.2.

Glava				
`tip`	`dolžina`	`podatki`		
4 bajti (ascii)	4 bajti (32 bitov)	*dolžina* (= 6) bajtov		
		16 bitov	16 bitov	16 bitov
MThd	⟨*dolžina*⟩	⟨*format*⟩	⟨*sledi*⟩	⟨*ritem*⟩

Tabela 2.2: Struktura glave.

MIDI datoteke so sestavljene iz glave in ene ali več sledi. Na voljo so trije različni formati:

- *format 0* - datoteka MIDI vsebuje eno sled, ki vsebuje vse podatke o notaciji in ritmu;
- *format 1* - datoteka MIDI vsebuje eno ali več sledi, ki so predvajane sinhronizirano; le prva sled vsebuje podatke o hitrosti predvajanja;
- *format 2* - datoteka MIDI vsebuje eno ali več sledi, ki so med seboj neodvisne.

Sled

Podatkovni del sledi vsebuje enega ali več dogodkov MIDI. Struktura sledi je prikazana v Tabeli 2.3.

Sled		
`tip`	`dolžina`	`podatki`
4 bajti (ascii)	4 bajti (32-bitov)	*dolžina* bajtov
MTrk	⟨*dolžina*⟩	⟨*dogodek*⟩...

Tabela 2.3: Struktura sledi.

Dogodki MIDI

Dogodki MIDI opisujejo celoten glasbeni zapis datoteke MIDI, od naslovov sledi do posameznih glasbenih dogodkov. Vsak dogodek vsebuje *časovno delto* (razliko), *tip dogodka* in tipu primerne podatke.

Časovna delta določa čas, ki mora preteči od sprožitve zadnjega dogodka. Če je delta 0 se dogodek sproži istočasno z zadnjim dogodkom. Dogodki, ki ne uporabljajo časa pojavitve, morajo biti na vrsti pred ostalimi. Na hitrost pojavitve dogodkov vpliva tempo posamezne sledi. Če ta ni določen, je privzeta vrednost 120 udarcev na minuto.

Obstajajo trije tipi dogodkov MIDI:

- *kontrolni dogodki MIDI* za sprožitev akcij MIDI,
- *sistemski izključujoči dogodki* za kontrolo strojne opreme MIDI,
- *meta dogodki* za določanje informacij o datoteki MIDI.

Kontrolni dogodki MIDI so najpomembnejši, saj določajo glasbene dogodke, kot so igranje tona in sprememba višine tona. Predstavljajo jedro glasbenega zapisa v datoteki MIDI. Vsak kontrolni dogodek je sestavljen iz časovne delte, kanala, ki mu je dogodek namenjen, tipa dogodka in posebnih parametrov [13]. Struktura kontrolnega dogodka je prikazana v Tabeli 2.4.

`Časovna delta`	`Tip dogodka`	`Kanal MIDI`	`Param. 1`	`Param. 2`
$\langle delta \rangle$	4 biti	4 biti	1 bajt	1 bajt

Tabela 2.4: Struktura kontrolnega dogodka MIDI.

Kontrolni dogodki se pošiljajo preko kanalov, ki jih zapisujejo v določeno sled. Za vsak kanal se določi instrument, katerega zvok je pripisan oddanim tonom. V Tabeli 2.5 je na voljo pregled kontrolnih dogodkov MIDI. Format MIDI podpira največ 16 (0-15) kanalov, vezanih na eno sled, s kanalom 10 rezerviranim za ritem sekcijo [14]. Za uporabo več kot 16 kanalov na enkrat je kanale potrebno porazdeliti po več različnih sledeh. Porazdelitev je uporabna predvsem z vidika lažjega urejevanja v urejevalnikih datotek MIDI.

Tip dogodka	Vrednost	Parameter 1	Parameter 2
Ustavitev tona	0x80	ton	moč
Sprožitev tona	0x90	ton	moč
Pritisk tona	0xA0	ton	pritisk
Upravljalnik	0xB0	številka upravljalnika	vrednost upravljalnika
Sprememba instrumenta	0xC0	številka instrumenta	*neuporabljen parameter*
Pritisk kanala	0xD0	pritisk	*neuporabljen parameter*
Upogibanje višine tona	0xE0	vrednost višine tona (LSB)	vrednost višine tona (MSB)

Tabela 2.5: Pregled kontrolnih dogodkov MIDI.

Osnovna dogodka sta sprožitev in ustavitev tona. Določata, kateri ton naj se začne ali preneha oddajati. MIDI ima na voljo 128 različnih tonov, od 0 do 127. Število 60 predstavlja *srednji C*. Pri sprožitvi parameter *moč* določa glasnost zaigranega tona, pri ustavitvi pa se ta parameter po navadi ignorira. V Tabeli 2.6 je prikazana struktura dogodka za sprožitev in ustavitev tona MIDI [15].

Ustavitev/Sprožitev tona	Kanal MIDI	Ton	Moč
0x80/0x90	0-15	0-127	0-127

Tabela 2.6: Ustavitev/Sprožitev tona MIDI.

2.3.2 Uporaba

Primer uporabe ukazov MIDI za oddajanje tonov prikazuje Koda 2.1. *Midi_msg* je struktura, ki vsebuje podatke, ki določajo tip dogodka MIDI in vrednosti po-

trebnih parametrov. Funkcija *setChannel* določi tip zvoka kanala MIDI. Funkcija *playNote* sproži oddajanje tona MIDI na določenem kanalu z določeno glasnostjo.

```
union {
    unsigned long word;
    unsigned char data[4];
} midi_msg;

void MidiEngine::setChannel(int channel, int instrument)
{
    midi_msg.data[0] = 0xC0|channel;
    midi_msg.data[1] = instrument;
    midi_msg.data[2] = 0;
    midi_msg.data[3] = 0;

    event_flag = midiOutShortMsg(device, midi_msg.word);
    if (event_flag != MMSYSERR_NOERROR) {
       qDebug() << "Warning: MIDI Output is not open.\n";
    }
}

void MidiEngine::playNote(int note, int channel, int velocity)
{
    midi_msg.data[0] = 0x90|channel;
    midi_msg.data[1] = note;
    midi_msg.data[2] = velocity;
    midi_msg.data[3] = 0;

    event_flag = midiOutShortMsg(device, midi_msg.word);
    if (event_flag != MMSYSERR_NOERROR) {
       qDebug() << "Warning: MIDI Output is not open.\n";
    }
}
```

Koda 2.1: Primer uporabe ukazov MIDI.

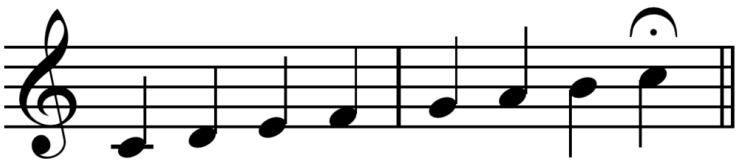
Poglavje 3

Glasbena teorija

3.1 Intervali

Glasbeni zapis nekega tona se imenuje *nota*. Ta določa višino tona in dolžino trajanja. V glasbeni teoriji je interval razdalja med dvema tonoma, oziroma razlika njunih višin. Če gre za dva zaporedna tona, je interval *melodičen*, pri dveh ali več sočasno zaigranih tonih pa gre za *harmoničen* interval [16].

Intervali so v zahodni glasbi razlika med dvema tonoma iz diatonične lestvice. Diatonična lestvica (Slika 3.1) je glasbena lestvica, sestavljena iz sedem zaporednih, po višini naraščajočih tonov in oktave (ponovitev prvega tona 12 poltonov višje) kot osmega tona. Med zaporednimi toni diatonične lestvice je pet velikih sekund ali *tonov* in dve mali sekundi ali *poltona* (glej Tabelo 3.1) [17].



Slika 3.1: Diatonična C-Dur lestvica.

Fizikalno gledano je interval razmerje med frekvencami tonov. Za oktavo velja razmerje 2:1. To pomeni, da je frekvenca višjega tona dva-krat višja od frekvence nižjega tona. Čeprav ljudje zaporedno višanje nekega tona za enak interval slišimo

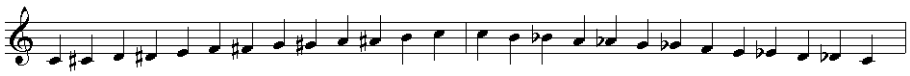
kot enakomerno, frekvenca tona narašča eksponentno. Intervale se zato meri tudi s *centi*, ki so enota, izpeljana iz logaritma razmerja višin (glej Izrek 3.1). Oktava je razdeljena na 1200 centov. V dvanajst-tonski tempirani uglasitvi, kjer imajo vsi poltoni enako velikost, je ta natanko sto centov [18].

Izrek 3.1 *Velikost intervala v centih od frekvence f_1 do f_2 je*

$$n = 1200 * log_2(\frac{f_2}{f_1}). \tag{3.1}$$

3.1.1 Glavni intervali

Kromatična lestvica je glasbena lestvica, sestavljena iz dvanajstih tonov. Vsak ton je za polton višje ali nižje od sosednjega tona (Slika 3.2). Ker so vsi poltoni enake velikosti, so toni kromatične lestvice enakomerno porazdeljeni [19].



Slika 3.2: Kromatična lestvica z začetkom pri tonu C.

V Tabeli 3.1 so prikazani najpogostejši intervali med posameznimi toni kromatične lestvice.

Interval	Poltoni	Okr.	Zveč./zmanj. intervali
Čista prima	0	č1	Zmanjšana sekunda
Mala sekunda (*polton*)	1	m2	Zvečana prima
Velika sekunda (*ton*)	2	v2	Zmanjšana terca
Mala terca	3	m3	Zvečana sekunda
Velika terca	4	v3	Zmanjšana kvarta
Čista kvarta	5	č4	Zvečana terca
(*tritonus*)	6		Zm. kvinta/Zv. kvarta
Čista kvinta	7	č5	Zmanjšana seksta
Mala seksta	8	m6	Zvečana kvinta
Velika seksta	9	v6	Zmanjšana septima
Mala septima	10	m7	Zvečana seksta
Velika septima	11	v7	Zmanjšana oktava
Čista oktava	12	č8	Zvečana septima

Tabela 3.1: Glavni intervali.

3.1.2 Konsonanca in disonanca

V glasbi se konsonanca (latinsko *consonare*, "zveneti skupaj") nanaša na glasbene konstrukte, kot so harmonija, akordi in intervali, za katere pravimo, da so *stabilni*. V splošnem stabilnost neke kombinacije tonov pomeni, da je sozvočje teh tonov za poslušalca prijetno. Nasprotje konsonance je disonanca. Nanaša se na kombinacijo tonov, ki za poslušalca ni prijetna in izzove občutek neugodja. Za disonančne intervale je značilno, da dajejo občutek napetosti, ki teži k razvezu (sprostitvi) v konsonančen interval.

Konsonanca je definirana z razmerji frekvenc tonov. Kot konsonančna se obravnava razmerja čim manjših naravnih števil. Najbolj konsonančen interval je tako prima (1:1), ki ji sledijo oktava (2:1), čista kvinta (3:2), čista kvarta (4:3), velika seksta (5:3), velika terca (5:4), mala terca (6:5) in mala seksta (8:5).

Konsonančne intervale delimo na popolne in nepopolne konsonance:

- popolne konsonance: čista prima, čista kvarta, čista kvinta, čista oktava;

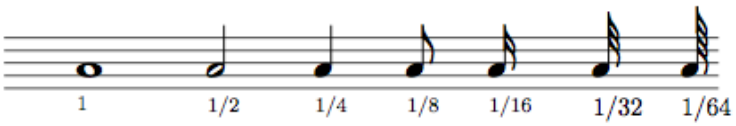
- nepopolne konsonance: mala terca, velika terca, mala seksta, velika seksta.

Vsi ostali intervali, z izjemo zmanjšane sekunde, spadajo med disonančne [20].

3.2 Ritem

Trajanje nekega tona je relativno glede na ostale tone. Absolutno trajanje je določeno šele z izbranim tempom in taktovskim načinom [21].

Dolžine trajanja not označujemo z različnimi oblikami (Slika 3.3) in poimenovanji (Tabela 3.2).



Slika 3.3: Trajanje not.

Ime	Relativno trajanje
Nota brevis	2
Celinka	1
Polovinka	1/2
Četrtinka	1/4
Osminka	1/8
Šestnajstinka	1/16
Dvaintridesetinka	1/32
Štiriinšestdesetinka	1/64
Stoosemindvajsetinka	1/128

Tabela 3.2: Relativno trajanje not.

Poglavje 4

Struktura aplikacije

4.1 Okolje

Celotna simulacija se odvija v 2D toroidnem mrežastem svetu velikosti $n * m$ celic. Populacijo predstavlja $0 < N < 16$ entitet, sicer pa je svet prazen.

4.2 Entiteta

Entiteta je model umetnega življenja in glavni akter v simulaciji. Predstavlja neko bitje, ki živi v 2D mrežastem svetu. Obnašanje entitet določajo *odzivi*. Ti določajo na kakšen dogodek, pod kakšnim pogojem in na kakšen način se entiteta odzove. Dogodke predstavljajo tako različne akcije sosednjih entitet kot tudi samo stanje sveta okoli posamezne entitete. V odziv na različne dogodke entitete oddajajo različne zvoke in se gibljejo po danem svetu. Pri načrtovanju je bila v oporo knjiga "Nature of Code" [22].

4.2.1 Struktura entitete

Entiteto sestavlja skupina parametrov. Del parametrov določa značilnosti entitet, kot so tip oddajanega zvoka, začetni položaj in odzivi. Ostali parametri beležijo trenutno stanje entitete in vrsto že zaigranih tonov.

Status pove, kakšno je trenutno stanje entitete. Ima dve možni vrednosti: *nedejaven* - entiteta ne oddaja nobenega zvoka in *aktiven* - entiteta v trenutnem

koraku oddaja zvok ali pa ima zapovedan premor. *Instrument* določa zvok zaigranih tonov. Možen je katerikoli od 128 instrumentov, ki jih MIDI ponuja. *Kanal* določa, kateri kanal MIDI pripada entiteti. V *trak* se zapisujejo vsi zaigrani toni entitete (seznam dogodkov MIDI). Trakovi so dolžine *število korakov*. Trakovi vseh entitet skupaj sestavljajo skladbo. *Trenutni ton* (lahko je premor) v primeru aktivnosti pove, kateri ton entiteta trenutno oddaja in v primeru nedejavnosti, kateri je bil nazadnje oddan. *Glasnost* določa glasnost oddanega tona. *Števec dob* določa, koliko dob (korakov) mora preteči, preden entiteta preneha z oddajanjem trenutnega tona. *Začetni položaj* določa položaj v 2D prostoru, ki ga entiteta zavzame pred začetkom izvajanja novih k korakov. *Trenutni položaj* določa trenutni položaj entitete. *Delta položaja* določa, v katero smer in za koliko korakov se bo entiteta premaknila ob koncu izvajanja trenutnega koraka. *Oceno* zaigranih tonov entiteta prejme po koncu izvajanja določenega števila (k) korakov. Na podlagi zaigrane sekvence in sekvenc ostalih entitet jo določi ocenitvena funkcija. *Stopnjo mutacije* parametrov entiteta prejme po koncu izvajanja k korakov. Določa verjetnost mutacije parametrov odzivov in opcijsko začetnega položaja. Vsaka entiteta vsebuje seznam možnih *odzivov* na dogodke. V posameznem koraku entiteta izvede le enega, tistega, katerega prioriteta je najvišja in je sprožitveni pogoj izpolnjen. V simulaciji je bil za prioriteto uporabljen razpon $[0, 7]$. Če ima več odzivov enako prioriteto, se sprožitveni pogoji preverjajo po vrstnem redu v seznamu.

4.3 Odziv

Odzivi določajo kdaj in kako se entiteta odzove na različne dogodke. Vsaka entiteta mora imeti vsaj en možen odziv, drugače bi bila v nedejavnem stanju skozi celotno simulacijo.

4.3.1 Struktura odziva

Vsak odziv ima osnovno zapovedano obliko. Poleg konstruktorja mora vsebovati pet osnovnih metod in tri osnovne parametre.

Osnovne metode

Sprožilo prejme nabor sosednjih entitet v radiju r in preveri, ali velja kateri od *sprožitvenih pogojev*, pri katerih se odziv sproži. Vrne *true*, če je pogojem zadoščeno in *false*, če pogojem ni zadoščeno. Uspešna sprožitev pomeni, da bo to odziv entitete v trenutnem koraku. *Generiranje tona* prejme nabor sosednjih entitet v radiju r. Ob sprožitvi odziva bo metoda vrnila ton, ki naj ga entiteta zaigra v trenutnem koraku. Ton določa vektor, ki vsebuje podatke o višini tona, glasnosti in trajanju. Lahko je premor. Metoda *generiranje premika* bo ob sprožitvi odziva vrnila smer in dolžino premika entitete. *Mutacija parametrov* prejme verjetnost mutacije *mut* in nato z verjetnostjo *mut* mutira vsak parameter odziva (osnovne in posebne). Če odziv vsebuje posebne parametre, ki pred začetkom izvajanja novih k korakov potrebujejo ponastavitev, se jih v metodi *reset* postavi na začetno vrednost.

Osnovni parametri

Prioriteta p določa, v kakšnem vrstnem redu se v posameznem koraku izvajanja simulacije preverjajo sprožitveni pogoji odzivov. *Radij zaznavanja* določa radij r okoli entitete. Entitete znotraj radija se obravnava kot *sosede*. V simulaciji je bil uporabljen razpon $[1, 5]$. *Delta premika* določa vrednost naslednjega premika po osi x in osi y.

Vsak parameter vsebuje minimalno in maksimalno vrednost. Ob mutaciji zavzame naključno vrednost iz nastavljenega razpona. Vsak odziv lahko poleg osnovnih parametrov vsebuje tudi lastne posebne parametre.

4.3.2 Delovanje

Vsaka entiteta ima odzive urejene po padajočem vrstnem redu prioritet. V vsakem koraku se po vrsti preverjajo sprožitveni pogoji odzivov. Prvi odziv, katerega sprožitvena funkcija vrne vrednost *true*, s funkcijama za generiranje tona in premika določi, s kakšnim tonom in premikom se entiteta odzove na trenutno stanje okoli nje. V vsakem koraku se lahko sproži največ en odziv posamezne entitete.

4.3.3 Mutacija

Vsak parameter, ki vpliva na delovanje odziva (radij zaznavanja, prioriteta, trajanje...) ima določen razpon vrednosti, ki jih lahko zavzame ($[Min, Max]$). Ob mutaciji zavzame naključno vrednost iz razpona.

4.3.4 Uporabljeni odzivi

Koncept odzivov je uporabljen z namenom enostavnega dodajanja novih odzivov, ki v simulacijo vnašajo poljubno stopnjo kompleksnosti. V predstavljeni verziji aplikacije so na voljo trije možni odzivi.

"Osamljeni" odziv

"Osamljeni" odziv se sproži, če v določenem radiju ni nobene druge entitete. Vrne privzeti ton z glasnostjo 100 in določenim trajanjem. Delta premika je določena z vrednostjo premika dx po osi x in vrednostjo premika dy po osi y. Dx in dy se spreminjata le z mutacijo, pri kateri zavzameta vrednost iz množice $\{-1, 0, 1\}$.

Oddaja *privzeti ton*, ki se spreminja le ob mutaciji in zavzame naključno vrednost iz razpona $[24, 96]$. *Trajanje* tona določa, koliko zaporednih korakov mora entiteta oddajati določen ton. Spreminja se z mutacijo in je določeno iz razpona $[1, 8]$.

"Akordni" odziv

"Akordni" odziv se sproži, če je v določenem radiju en sam sosed. Če sosed trenutno ni tiho, "akordni" odziv odda ton z glasnostjo 100 in določenim trajanjem, ki je za interval višje ali nižje od tona, ki ga oddaja sosed. Števec predznakov določa indeks vrednosti v seznamu predznakov, ki določi, ali se interval sosedovemu tonu prišteje ali odšteje (množenje intervala z 1 ali -1). Koda 4.1 prikazuje generiranje oddanega tona "akordnega" odziva. Delta premika je določena z vrednostjo premika dx po osi x in vrednostjo premika dy po osi y. Dx in dy se spreminjata le z mutacijo, pri kateri zavzameta vrednost iz množice $\{-1, 0, 1\}$.

Parameter *število korakov* določa število potrebnih predznakov. Vsebuje ⟨*število korakov*⟩ *predznakov*, ki za posamezno sprožitev določajo, ali bo ton odziva za interval višje (interval pomnožen z 1) ali nižje (interval pomnožen z -1) od

referenčnega tona. *Števec predznakov* se povečuje z vsako sprožitvijo in določa, kateri predznak se upošteva pri trenutnem odzivu. Po koncu izvajanja vsakih k korakov se postavi nazaj na vrednost 0. *Interval* določa uporabljen interval iz razpona $[0, 12]$. *Trajanje* tona določa, koliko zaporednih korakov mora entiteta oddajati določen ton. Določeno je iz razpona $[1, 8]$.

```
int tone;
int sign = signs.at(sign_counter++);
int neighbour_tone = neighbour->getCurrentTone();

if (neighbour_tone != midi_state::PAUSE) {
    tone = neighbour_tone + (sign * interval.getValue());

    if (tone < 24) tone += 12;
    else if (tone > 96) tone -= 12;
}
else tone = midi_state::PAUSE;
```

Koda 4.1: Generiranje tona "akordnega" odziva.

"Tihi" odziv

"Tihi" odziv je enak kot "osamljeni" odziv, le da namesto privzetega tona vedno zapove premor.

4.4 Generator naključnih števil

Za generiranje naključnih števil je uporabljen algoritem *Marsenne Twister* iz knjižnice Boost. Aplikacija omogoča uporabo lastnega semena. Ob isti nastavitvi parametrov in uporabi istega semena bo tako potek simulacije vedno enak.

4.5 Poganjalnik MIDI

Poganjalnik MIDI skrbi za začetek in prenehanje oddajanja zvokov entitet. Pri inicializaciji simulacije se določi, kateri kanal MIDI pripada kateri entiteti. Določijo se tudi instrumenti, ki določajo tip zvoka posameznih entitet.

Vrednosti MIDI

Z vrednostmi MIDI se beležijo zaporedja dogodkov MIDI. *Toni* so predstavljeni kot naravna števila v razponu od 0 do 127. Razdalja med sosednjima številkama predstavlja polton. Določeni sta vrednosti za *premor*, ki pomeni tišino in *vzdrževanje*, ki pomeni vzdrževanje oddajanja tona iz prejšnje dobe. Lahko traja več dob. Tako se beleži različno trajanje tonov.

Po vsakem koraku se v trak vsake entitete zapiše nova vrednost MIDI. Na koncu trakovi vseh entitet ene generacije skupaj sestavljajo skladbo (Tabela 4.1).

Koraki / Trakovi	1	2	...	k
Trak entitete 1	v_{11}	v_{12}	...	v_{1k}
Trak entitete 2	v_{21}	v_{22}	...	v_{2k}
⋮				
Trak entitete N	v_{N1}	v_{N2}	...	v_{Nk}

Skladba

Tabela 4.1: Trakovi entitet skupaj sestavljajo skladbo.

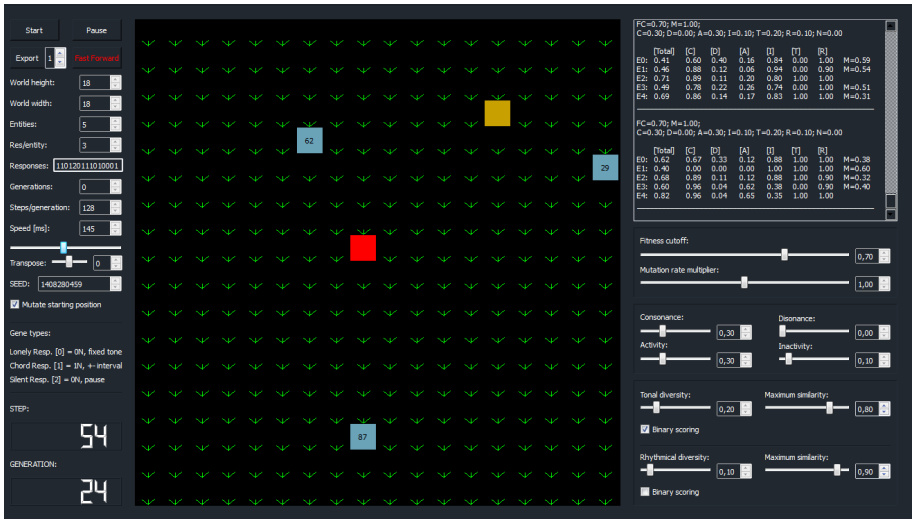
Poganjalnik MIDI omogoča izvoz skladbe zadnje generacije entitet. Vse trakove entitet se združi v skladbo in izvozi v formatu MIDI (.mid).

4.6 Grafični uporabniški vmesnik

Grafični uporabniški vmesnik (Slika 4.1) omogoča uporabniku kontrolo nad parametri simulacije. Omogoča nastavitev začetnih parametrov, določanje hitrosti simulacije, transpozicijo tonov, kontrolo nad cenilko in mutacijo in možnost izvoza produciranih tonov v formatu MIDI (.mid). Pri izvozu je možna nastavitev več ponovitev.

Na sredini je prikazan svet, v katerem živijo entitete. Barva entitet sporoča njihovo stanje. Modra za aktivne entitete, ki oddajajo neki ton, rumena za aktivne, ki prestajajo premor in rdeča za neaktivne entitete, brez v trenutnem koraku aktivnega odziva. V desnem zgornjem oknu so prikazane uteži ocenitvene funkcije,

ocene in stopnja mutacij entitet zadnje generacije. V levem spodnjem kotu sta prikazana trenutni korak in generacija.



Slika 4.1: Grafični uporabniški vmesnik.

4.7 Jedro

Jedro skrbi za inicializacijo vseh entitet in parametrov in potek celotne simulacije.

4.7.1 Parametri simulacije

Parametri simulacije določajo osnovno obliko simulacije. Preko grafičnega uporabniškega vmesnika uporabnik pred začetkom simulacije določi vrednost parametrov, kot so velikost sveta, število in struktura entitet. Po inicializaciji oblike ni več mogoče spreminjati. Uporabniku ostane kontrola nad hitrostjo izvajanja simulacije in parametri ocenitvene funkcije. Seznam parametrov simulacije je prikazan v Tabeli 4.2.

`Parameter`	`Razpon vrednosti`
Višina sveta	$0 < n$
Širina sveta	$0 < m$
Število entitet	$0 < N < 16$
Število odzivov na entiteto	$0 < o$
Niz indeksov odzivov	$dolžina\ niza = N * o$
Število generacij	$0 \leq g; 0 = \infty$
Število korakov	$0 < k$
Hitrost simulacije (premor med koraki [ms])	$0 < h \leq 300$
Transpozicija tonov	$-24 \leq t \leq 24$
Mutacija začetnih položajev entitet	$true/false$
Izvoz zadnje sekvence	$0 <$ *število ponovitev*
Prag ustreznosti	$0 \leq p \leq 1$
Stopnja mutacije	$0 \leq mut \leq 1$
Koeficienti ocenitvene funkcije	$0 \leq c \leq 1$
Največja podobnost tonov	$0 < ts \leq 1$
Največja podobnost trajanja tonov	$0 < rs \leq 1$
Binarno ocenjevanje melodične raznolikosti	$true/false$
Binarno ocenjevanje ritmične raznolikosti	$true/false$

Tabela 4.2: Parametri simulacije.

4.7.2 Inicializacija

V jedru se najprej inicializira parametre simulacije, nato pa še entitete, ki se jim dodelita vnaprej določeno število odzivov in začetni položaj. Parametri odzivov zavzamejo naključno vrednost iz vnaprej določenih razponov vrednosti. Odzivi so lahko dodeljeni naključno iz nabora odzivov ali pa jih dodeli entitetam uporabnik z nizom indeksov (glej Tabelo 4.3).

Odziv	Indeks
“Osamljeni” odziv	0
“Akordni” odziv	1
“Tihi” odziv	2

Tabela 4.3: Indeksi odzivov.

4.7.3 Potek simulacije

V vsakem koraku simulacije se preveri stanje vsake entitete. Če je entiteta v neaktivnem stanju se po vrsti preverja sprožitvene pogoje njenih odzivov. Ob sprožitvi odziv določi morebitni oddani ton in premik. Po izvedbi k korakov se zaigrane sekvence entitet oceni glede na sekvence ostalih entitet. Na podlagi ocen se mutira njihove parametre, nato pa se vse začasne vrednosti ponastavi in entitete pripravi na novih k korakov. Koda 4.2 prikazuje potek simulacije.

```
while (simulation_state::RUNNING) {

    step_counter++;
    processNextStep();

    if (step_counter == steps_per_generation) {

        evaluateEntities();
        mutateEntities();

        resetEntities();
        step_counter = 0;
        generation_counter++;

    }
}
```

Koda 4.2: Potek simulacije.

Procesiranje novega koraka

V zanki se posodobi stanje vseh entitet. Če $status = aktiven$ in $števec\_dob > 0$, stanje entitete ostane nespremenjeno. V primeru, da $status = aktiven$ in $števec\_dob = 0$ ali $status = neaktiven$, entiteta preneha z morebitnim oddajanjem tona. Po prioritetnem vrstnem redu se preverjajo sprožitveni pogoji odzivov. Ob sprožitvi odziv določi nov ton ali premor, trajanje in premik. Po obdelavi vseh entitet, tiste z določeno akcijo pričnejo oddajati ton in izvedejo premik. Nov oddani ton, vzdrževanje tona ali premor se zapiše v trak entitet. Entitete, pri katerih ni bil sprožen noben odziv, postanejo neaktivne.

Ocena in mutacija

Ob koncu izvedbe k korakov se zaigrano sekvenco vsake entitete oceni glede na skladbo kot celoto. Glede na prejeto oceno in željeno stopnjo mutacije se mutira parametre odzivov in opcijsko tudi začetni položaj entitet. Celoten postopek se nato ponovi.

Križanje

Preizkušeno je bilo tudi iskanje rešitve z operacijo križanja. Starša nadomestita dva otroka, ki vsebujeta delež odzivov vsakega starša. Ta metoda iskanja se je, tudi v kombinaciji z mutacijami, izkazala za neuspešno, saj s prevelikimi posegi v sodelovanje entitet simulacijo oddaljuje od rešitve.

Poglavje 5

Algoritem

5.1 Cilj

Z uporabo usmerjenega iskanja se išče zaporedje aktivnosti, ki jih mora posamezna entiteta izvesti, tako da bo ustvarjena kombinacija tonov N entitet zvenela poslušljivo.

5.2 Iskanje rešitve

Usmerjanje iskanja temelji na naključnem iskanju po področju rešitev, ki se manjša s kvaliteto rešitve. Naključno iskanje predstavlja mutacija parametrov odzivov entitet. Razpon, po katerem se išče rešitev, se manjša z verjetnostjo mutacije, ki je za posamezno entiteto manjša, boljša, kot je njena ocena ob koncu izvajanja določenega števila korakov (5.1). Stopnjo mutacije določi uporabnik preko grafičnega vmesnika.

$$Verjetnost\_mutacije = (1 - Ocena) * Stopnja\_mutacije$$

pri čemer velja

$$Verjetnost\_mutacije \in [0, 1]. \tag{5.1}$$

Kot tehnična opora in velika inspiracija pri načrtovanju algoritma je služila

knjiga “Computational Beauty of Nature” [23].

5.3 Ocena

Vrste zaigranih tonov (trak) entitet se ocenjuje glede na skladnost s toni drugih entitet, razmerje med zaigranimi toni in premori, ter ritmično in melodično raznolikost. Končna ocena je sestavljena iz uteženih ocen osnovnih vidikov, s katerimi ocenjujemo poslušljivost neke kombinacije tonov. Vidiki ocene so *konsonanca*, *disonanca*, *aktivnost*, *neaktivnost*, *melodična raznolikost* in *ritmična raznolikost*.

Entitete se ustvarjanja celotne skladbe ne zavedajo, zato ne morejo identificirati in prilagoditi vrednosti parametra, ki je krivec za neskladje. Glede na podano oceno bolj ali manj agresivno preiskujejo prostor možnih konfiguracij svojih odzivov.

Uporabnik določi prag ustreznosti, ki določa najmanjšo dovoljeno oceno entitet. Entitete z večjo ali enako oceno ob koncu izvajanja vsakih k korakov niso podvržene mutaciji. Z nižanjem pragu ustreznosti se entitetam dovoli nekaj svobode pri ustvarjanju, kar lahko pripelje do zanimivih sekvenc, ki sicer niso popolnoma skladne z ocenitveno funkcijo.

5.3.1 Konsonanca

S konsonančno oceno se nagrajuje entitete, katerih oddani toni so v času trajanja skladni (v konsonančnem intervalu) s toni ostalih entitet. V producirano glasbo se želi vnesti stabilnost in harmonijo.

Trak entitete se v vseh korakih primerja s trakovi ostalih entitet. Če je entiteta v nekem koraku oddajala neki ton, se preveri, kakšne intervale v tem koraku tvori z morebitnimi oddajanimi toni drugih entitet. Za vsako entiteto se beležita dve vrednosti: število konsonančnih intervalov ($kons$) in število primerjanih intervalov ($intervali$). Njuno razmerje predstavlja oceno konsonance (5.2).

$$Ocena\_konsonance = \begin{cases} \frac{kons}{intervali}, & 0 < intervali, \\ 0, & \text{sicer.} \end{cases} \tag{5.2}$$

5.3.2 Disonanca

Z disonančno oceno se želi v glasbo vnesti nekaj napetosti.

Podobno kot pri preverjanju konsonance se beleži število disonančnih intervalov (*dis*) in število primerjanih intervalov (*intervali*). Njuno razmerje predstavlja oceno disonance (5.3).

$$Ocena\_disonance = \begin{cases} \frac{dis}{intervali}, & 0 < intervali, \\ 0, & \text{sicer.} \end{cases} \tag{5.3}$$

5.3.3 Aktivnost

Z oceno aktivnosti se lahko entitete spodbudi k produciranju tonov, saj so neaktivne entitete kaznovane.

Za oceno aktivnosti se beleži, v koliko korakih simulacije je entiteta oddajala neki ton (*aktivnost*). Oceno predstavlja razmerje med aktivnostjo in številom korakov (k) (5.4).

$$Ocena\_aktivnosti = \frac{aktivnost}{k} \tag{5.4}$$

5.3.4 Neaktivnost

Ob večjem številu entitet in njihovi aktivnosti se lahko neka smiselna struktura skladbe izgubi v prevelikem številu hkrati zaigranih tonov. Z oceno neaktivnosti se kaznuje preveč aktivne entitete. Želi se doseči primerno razmerje med aktivnostjo in neaktivnostjo entitet.

Oceno neaktivnosti, podobno kot oceno aktivnosti, predstavlja razmerje med številom korakov, v katerih entiteta ni oddajala nobenega tona (*neaktivnost*) in številom vseh korakov (k) (5.5).

$$Ocena\_neaktivnosti = \frac{neaktivnost}{k} \tag{5.5}$$

5.3.5 Melodična raznolikost

Cilj melodične raznolikosti je, da se posamezne entitete sili v oddajanje bolj razgibanih sekvenc tonov. Izogniti se želi monotonosti skladbe.

Za vsak različni ton v traku entitete se zabeleži frekvenca pojavitve (5.6).

$$frekvenca\_tona = \frac{stevilo\_pojavitev\_tona}{stevilo\_vseh\_zaigranih\_tonov} \tag{5.6}$$

Uporabnik preko grafičnega vmesnika določi največjo dovoljeno podobnost tonov in opcijsko binarno ocenjevanje melodične raznolikosti. *Največja podobnost tonov* ($0 < ts \leq 1$) določa največjo tolerirano frekvenco pojavitve posameznega tona. Bolj, kot jo frekvence tonov presegajo, slabša je ocena. *Binarno ocenjevanje melodične raznolikosti* (t_bin) določa način ocenjevanja. Zavzame lahko dve vrednosti: *true* - če katera od frekvenc preseže mejo ts, entiteta prejme oceno 0 za melodično raznolikost in *false* - ocenjevanje poteka po algoritmu 5.3.5.

Algoritem za ocenjevanje melodične raznolikosti

Ocena pada glede na stopnjo preseganja največje podobnosti tonov. Ko se ne ocenjuje binarno je ts hkrati tudi najnižja možna ocena, ki jo posamezna entiteta lahko prejme (5.7). Koda 5.1 prikazuje izračun ocene melodične raznolikosti.

$$Ocena = 1 - (frekvenca\_tona - ts) = 1 - (1 - 0.2) = 0.2 \geq ts \tag{5.7}$$

```
double ocena_melodicne_raznolikosti = 1.0;

foreach (double frekvenca_tona, frekvence_tonov) {

    if (frekvenca_tona > ts && t_bin) {
        ocena_melodicne_raznolikosti = 0;
        break;
    }
    ocena_melodicne_raznolikosti -= frekvenca_tona > ts
                                  ? frekvenca_tona - ts
                                  : 0.0;
}
```

Koda 5.1: Izračun ocene melodične raznolikosti.

5.3.6 Ritmična raznolikost

Z oceno ritmične raznolikosti se želi doseči, da so producirane sekvence entitet čim bolj ritmično razgibane in za poslušalca zanimive.

Zabeleži se frekvence trajanj tonov v traku entitete (5.8). Za razliko od ocenjevanja melodične raznolikosti se trajanje zadnjega tona v traku ne upošteva, saj je lahko zaradi omejenega števila korakov izvajanja odrezan.

$$frekvenca\_trajanja = \frac{stevilo\_pojavitev\_trajanja}{stevilo\_vseh\_v\_celoti\_zaigranih\_tonov} \tag{5.8}$$

Uporabnik preko grafičnega vmesnika določi največjo podobnost trajanja tonov in opcijsko binarno ocenjevanje ritmične raznolikosti. *Največja podobnost trajanja tonov* ($0 < rs \leq 1$) določa največjo tolerirano frekvenco pojavitve dolžine trajanja tonov. Bolj, kot jo frekvence presegajo, slabša je ocena. *Binarno ocenjevanje ritmične raznolikosti* (r_bin) določa način ocenjevanja. Zavzame lahko dve vrednosti: *true* - če katera od frekvenc preseže mejo rs, entiteta prejme oceno 0 za ritmično raznolikost in *false* - ocenjevanje poteka po algoritmu 5.3.6.

Algoritem za ocenjevanje ritmične raznolikosti

Ocena pada glede na stopnjo preseganja največje frekvence trajanja tonov. Ko se ne ocenjuje binarno je rs hkrati tudi najnižja možna ocena, ki jo posamezna entiteta lahko prejme (5.9). Koda 5.2 prikazuje izračun ocene ritmične raznolikosti.

$$\begin{aligned} Odbitek_1 &= frekvenca\_trajanja_1 - rs = 0.6 - 0.3 = 0.3 \\ Odbitek_2 &= frekvenca\_trajanja_2 - rs = 0.4 - 0.3 = 0.1 \\ Ocena &= 1 - Odbitek_1 - Odbitek_2 = 1 - 0.3 - 0.1 = 0.6 \geq rs \end{aligned} \tag{5.9}$$

```
double ocena_ritmicne_raznolikosti = 1.0;

foreach (double frekvenca_trajanja, frekvence_trajanj) {

    if (frekvenca_trajanja > rs && r_bin) {
        ocena_ritmicne_raznolikosti = 0;
        break;
    }
```

```
    ocena_ritmicne_raznolikosti -= frekvenca_trajanja > rs
                                   ? frekvenca_trajanja - rs
                                   : 0.0;
}
```

Koda 5.2: Izračun ocene ritmične raznolikosti.

5.3.7 Ocenitvena funkcija

Končno oceno entitete sestavlja utežena vsota vidikov ocene:

$$\begin{aligned} Ocena = {} & Utez\_konsonance * Ocena\_konsonance \\ & + Utez\_disonance * Ocena\_disonance \\ & + Utez\_aktivnosti * Ocena\_aktivnosti \\ & + Utez\_neaktivnosti * Ocena\_neaktivnosti \\ & + Utez\_melodicnosti * Ocena\_melodicne\_raznolikosti \\ & + Utez\_ritmicnosti * Ocena\_ritmicne\_raznolikosti \\ & + Nevtralna\_utez \end{aligned}$$

pri čemer velja

$$Ocena \in [0, 1]$$

in

$$Utezi \in [0, 1]$$

ter

$$Nevtralna\_utez = 1 - \sum Utezi. \tag{5.10}$$

Poglavje 6

Uporaba in rezultati

6.1 Začetni parametri

Entitete si lahko predstavljamo kot glasbenike, ki se jih poskuša pripraviti do sodelovanja. Čeprav je število glasbenikov omejeno na največ 15, postane hitro jasno, da bo glasba ob prevelikem številu aktivnih in ne nujno popolnoma usklajenih glasbenikov prenasičena in brez jasne oblike.

6.1.1 Velikost okolja

Izkazalo se je, da premajhna površina okolja pripelje do neaktivnosti entitet. Pri testiranju je bil za radij zaznavanja uporabljen razpon $[1, 5]$. Pri več kot treh entitetah tako okolje, manjše od $15 * 15$ ni primerno, saj se zaradi prevelikega števila sosedov ne aktivira nobeden od odzivov. Največja še smiselna velikost je $30 * 30$. Za višino in širino se je kot optimalen izkazal razpon $[15, 25]$, saj se tako glede na izbran razpon radija zaznavanja doseže najboljša kombinacija osamljenosti in skupin entitet. Ob večjem razponu radija bi dosegli dobre rezultate tudi z večjo površino okolja.

6.1.2 Število korakov

Večje, kot je število korakov, večje je področje rešitev in težje je priti do dobrega rezultata. Testiranje je bilo opravljeno s 512, 256, 128 in 64 koraki. Najboljše

rešitve so bile najdene pri največ 128 korakih.

6.1.3 Odzivi in število entitet

Pri testiranju se je izkazalo, da veliko vlogo pri izboru števila entitet igra število in tip odzivov, ki se jih entitetam dodeli. Predpostavimo, da se entitetam pripiše le tipe odzivov, ki oddajajo neki ton (ne oddajajo premora).

Veliko odzivov

Več pripisanih odzivov pomeni, da bo posamezna entiteta z večjo verjetnostjo našla primerno akcijo in oddala ton, kar ob velikem številu entitet pomeni prenasičenost glasbe. Na voljo je zato "tihi" odziv, s katerim se v skladbo vnesejo premori kljub velikemu številu odzivov in veliko entitetam. Pri manjšem številu entitet ta problem ne pride do izraza, saj tudi ob veliki aktivnosti težko pride do prenasičenosti.

Malo odzivov

Pri manjšem številu pripisanih odzivov in s tem manjšo verjetnostjo, da bodo vse entitete aktivne v vsakem koraku, je lahko uporabljeno večje število entitet. Uporabljeni odzivi se sprožajo glede na število sosedov v radiju zaznavanja. Pri večjem številu entitet je večja verjetnost, da bo nekaj entitet imelo preveč sosedov za sprožitev kateregakoli odziva. Aktivnost entitet tako pada obratno sorazmerno s številom entitet, ko imajo te dodeljeno majhno število odzivov.

Optimalne kombinacije števila entitet in odzivov

Vsaka konfiguracija ne pripelje do primernega rezultata. Izkazalo se je, da kombinacije od pet do sedem entitet s tremi ali štirimi odzivi najpogosteje pripeljejo do poslušljivih, ne prenasičenih glasbenih sekvenc.

Optimalne kombinacije odzivov

Na rezultat vplivajo tudi uporabljeni odzivi. Če je uporabljenih preveč "osamljenih" odzivov se dobi nezanimive monotone sekvence. Ob uporabi prevelikega

števila “akordnih” odzivov postane sekvenca preveč nestabilna. Ob prevelikem številu “tihih” odzivov pa lahko sekvenca hitro postane prazna.

Najbolj uspešne so uravnotežene kombinacije “osamljenih” in “akordnih” odzivov z manjšim številom entitet, ki vsebujejo največ en “tihi” odziv. V Tabeli 6.1 je prikazanih nekaj uspešnih kombinacij za pet entitet s tremi ali štirimi odzivi.

Št.odzivov	Ent.1	Ent.2	Ent.3	Ent.4	Ent.5
3	011	012	111	001	001
3	011	112	111	000	002
4	0011	0011	0111	0112	1112
4	0112	0011	1111	0112	0112

Tabela 6.1: Uspešne kombinacije odzivov.

6.2 Strategija iskanja

Področje rešitev je ogromno, zato je pri iskanju rešitve pomemben pristop in kakšne zahteve se postavi entitetam.

Z utežmi se določi pomembnost posameznega vidika ocenitvene funkcije, kar vpliva na končno obliko rešitve. Iskanja se lahko loti na različne načine. Lahko se začne z velikim poudarkom na konsonanci in se šele, ko je dosežena visoka konsonančnost tonov, začne z dodajanjem ostalih vidikov ocene. Seveda je lahko že od začetka funkcija zastavljena kar se da strogo in kompleksno.

Veliko vlogo pri iskanju rešitve igra prag ustreznosti. Kljub precej kompleksni ocenitveni funkciji, ki omogoča oblikovanje rešitve, je še vedno težko definirati, kaj natančno naredi neko glasbo zanimivo. V primeru te simulacije je lahko z nižanjem pragu ustreznosti dopuščena določena mera kaosa, ki lahko na zanimive načine popestri slišano.

Tako kot s pragom ustreznosti se lahko na potek iskanja vpliva tudi s stopnjo mutacije. V primeru, da simulacija producira glasbo, ki je sicer skladna z ocenitveno funkcijo, a nevšečna, se lahko z večanjem stopnje mutacije poskrbi za iskanje po večjem razponu rešitev. Podobno se, ko se bliža želenemu rezultatu, stopnjo mutacije lahko pomanjša. Verjetnost, da se simulacija spet oddalji od rešitve, je

tako manjša.

Nekaj v testiranju uspešnih konfiguracij je prikazanih v Tabeli 6.2.

Konfiguracije / Parametri	`Konfiguracija 1`	`Konfiguracija 2`
`Prag ustreznosti`	0.70	0.75
`Konsonanca`	0.30	0.30
`Disonanca`	0	0.05
`Aktivnost`	0.30	0.20
`Neaktivnost`	0.10	0.10
`Melodičnost`	0.20; $ts = 0.80$; (bin.)	0.25; $ts = 0.60$;
`Ritmičnost`	0.10; $rs = 0.90$	0.10; $rs = 0.90$; (bin.)
Višina sveta	18	25
Širina sveta	18	25
Število entitet	5	10
Število odzivov	3	3

Tabela 6.2: Primer uspešnih konfiguracij.

6.3 Rezultati

Simulacijo sestavljajo entitete, ki se celotne "slike" ne zavedajo. Vsaka ima nabor enostavnih odzivov, ki producirajo en ton na enkrat. Ob načrtovanju poteka simulacije so bila tako pričakovanja nizka. Gotovo bo rezultat zanimiv, a računalniško obarvan...

Vendar so rezultati presegli vsa pričakovanja. Čeprav je veliko ustvarjenih sekvenc zanimivih le z vidika ustreznosti z ocenitveno funkcijo, ima zadosti sekvenc kvalitete pravih kompleksnih kompozicij. Sestavlja jih velika melodična in ritmična raznolikost tonov s ponavljajočimi se deli osnovne teme, prehodi in premori. Ti ob dodatku zvoka koncertnega klavirja pričarajo občutek pravih klavirskih koncertov. Primeri so zbrani na spletnem repozitoriju [24].

Poglavje 7

Sklepne ugotovitve

7.1 Pristop

Na področju algoritmičnega generiranja glasbe je veliko možnih pristopov. Algoritmi večinoma analizirajo in počasi usmerjeno prilagajajo začetno sekvenco tonov. Opisan pristop se razlikuje v tem, da za generiranje tonov uporablja model umetnega življenja (entiteta), ki se ustvarjanja glasbe ne zaveda in ni zmožen specifičnega prilagajanja lastnih parametrov. Izkazalo se je, da je skupino entitet, s pomočjo ocenjevanja in mutacij, možno pripeljati do sodelovanja in produciranja smiselnih in poslušljivih glasbenih sekvenc.

Ustvarjen je algoritem, ki z uporabo usmerjenega iskanja išče zaporedje aktivnosti, ki jih morajo entitete izvesti, tako da bodo v primeru N entitet producirani toni sestavljali poslušljivo celoto.

7.2 Uporabnost

Producirane sekvence so uporabne kot ideje pri skladanju lastnih kompozicij. Nekatere so zaradi visoke kvalitete in kompleksnosti že same po sebi poslušljive skladbe.

7.3 Nadgradnja

Algoritem bi lahko bil nadgrajen z uporabo mehke logike za boljše preiskovanje prostora rešitev. Uporabniku bi lahko bila omogočena še večja kontrola nad parametri odzivov in načinom preiskovanja.

Ob ustvarjenem naboru primernih sekvenc bi lahko namenski algoritem te s pomočjo ocenitvene funkcije sestavil v večjo, smiselno celoto. Mogoč bi bil tudi razsek sekvenc in uporaba manjših smiselnih delov kot gradnikov.

Literatura

[1] J. D. F. Rodriguez and F. J. Vico, "AI methods in algorithmic composition: A comprehensive survey," *CoRR*, vol. abs/1402.0585, 2014. Dostopno na: http://arxiv.org/abs/1402.0585

[2] G. Papadopoulos and G. Wiggins, "A genetic algorithm for the generation of jazz melodies," in *PROCEEDINGS OF STEP 98*, 1998, pp. 7–9. Dostopno na: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.5.3295

[3] E. R. Miranda, S. Kirby, and P. M. Todd, "On computational models of the evolution of music: From the origins of musical taste to the emergence of grammars," 2003. Dostopno na: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.161.2629

[4] G. Papadopoulos and G. Wiggins, "Ai methods for algorithmic composition: A survey, a critical view and future prospects," in *IN AISB SYMPOSIUM ON MUSICAL CREATIVITY*, 1999, pp. 110–117. Dostopno na: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.3.8064

[5] P. S. Langston, "Six techniques for algorithmic music composition," 1988. Dostopno na: http://www.langston.com/Papers/amc.pdf

[6] E. R. Miranda, "Voices of artificial life: On making music with computer models of nature." Dostopno na: http://www.academia.edu/3670192/Voices_of_Artificial_Life_On_Making_Music_with_Computer_Models_of_Nature

[7] ——. Dostopno na: http://neuromusic.soc.plymouth.ac.uk/

[8] ——, *Readings in Music and Artificial Intelligence (Contemporary Music Studies)*. Routledge, 2000.

[9] Wikipedia, "Qt — Wikipedia, the free encyclopedia." Dostopno na: http://en.wikipedia.org/wiki/Qt_(software)

[10] ——, "Boost c++ library — Wikipedia, the free encyclopedia." Dostopno na: http://en.wikipedia.org/wiki/Boost_(C++_libraries)

[11] ——, "Midi — Wikipedia, the free encyclopedia." Dostopno na: http://en.wikipedia.org/wiki/MIDI

[12] C. S. Sapp, "Outline of the standard midi file structure," The Stanford Center for Computer Research in Music and Acoustics. Dostopno na: http://improv.sapp.org/doc/misc/MidiFileFormat.html

[13] "Midi," Midi Manufacturers Association. Dostopno na: http://www.midi.org/aboutmidi/intromidi.pdf

[14] R. Stansifer, "Midi file structure," Florida Institute of Technology. Dostopno na: http://cs.fit.edu/~ryan/cse4051/projects/midi/midi.html

[15] "Midi," Sonicspot. Dostopno na: http://www.sonicspot.com/guide/midifiles.html

[16] Wikipedia, "Interval — Wikipedia, the free encyclopedia." Dostopno na: http://en.wikipedia.org/wiki/Interval_(music)

[17] ——, "Diatonic scale — Wikipedia, the free encyclopedia." Dostopno na: http://en.wikipedia.org/wiki/Diatonic_scale

[18] ——, "Cent — Wikipedia, the free encyclopedia." Dostopno na: http://en.wikipedia.org/wiki/Cent_(music)

[19] ——, "Chromatic scale — Wikipedia, the free encyclopedia." Dostopno na: http://en.wikipedia.org/wiki/Chromatic_scale

[20] ——, "Consonance and dissonance — Wikipedia, the free encyclopedia." Dostopno na: http://en.wikipedia.org/wiki/Consonance_and_dissonance

[21] ——, "Rhythm — Wikipedia, the free encyclopedia." Dostopno na: http://en.wikipedia.org/wiki/Rhythm

[22] D. Shiffman, *The Nature of Code.* Samozaložba, 2012. Dostopno na: http://natureofcode.com/book/

[23] G. W. Flake, *The Computational Beauty of Nature: Computer Explorations of Fractals, Chaos, Complex systems, and Adaptation.* The MIT Press, 1998.

[24] A. Placer, "Take artificial five - music sample repository." Dostopno na: https://soundcloud.com/take-artificial-five